ulm university universität
uulm

Institut für Angewandte Informationsverarbeitung
Dr. Andreas F. Borchert

2. Dezember 2014
Blatt 8

Objektorientierte Programmierung mit C++ (WS 2014/2015)

Abgabe bis zum 9. Dezember 2014, 14:00 Uhr

Lernziele:

- Entwicklung einer Klasse mit Ausnahmenbehandlungen
- Umsetzung des RAII-Prinzips

Aufgabe 10: Speicherobjekte

Die POSIX-Schnittstelle bietet mit *mmap* eine Schnittstelle an, mit der u.a. Dateien in den eigenen Adressraum abgebildet werden können. Da diese Abbildungen nach Gebrauch mit *munmap* wieder rückgängig zu machen sind, bietet es sich an, so etwas in eine geeignete Klasse zu verpacken, die das RAII-Prinzip umsetzt.

Im Rahmen dieser Aufgabe ist eine Klasse *MemObject* zu entwickeln, die entsprechend des RAII-Prinzips

- bei der Konstruktion die Parameter für *mmap* entgegennimmt und *mmap* aufruft,
- beim Abbau die Abbildung wieder rückgängig macht mit Hilfe von *munmap*, falls *mmap* erfolgreich war und
- auf die abgebildete Speicherfläche im Erfolgsfalle einen geeigneten Zugang gewährt.

Der Konstruktor sollte mit Ausnahme von *addr* alle Parameter von *mmap* unterstützen. So könnte beispielsweise die Datei */usr/dict/words* in den Speicher abgebildet werden:

```
#include <fcntl.h>
#include <sys/mmap.h>
#include <unistd.h>
#include "MemObject.hpp"

int main() {
    // ...
```

```
    int fd = open("/usr/dict/words", O_RDONLY);
    if (fd > 0) {
        off_t res = lseek(fd, 0, SEEK_END);
        if (res >= 0) {
            size_t len = res;
            MemObject words{fd, /* offset = */ 0, /* size = */ len,
                PROT_READ, MAP_SHARED};
            // work on words
        }
    }
}
```

Achten Sie darauf, dass Ihre Ressource nicht einfach kopiert werden kann, d.h. eine einfache Zuweisung ist zu unterbinden, aber ein *move assignment* oder ein *move constructor* wäre realisierbar.

Dabei sind Ausnahmen zu erzeugen, wenn ein Zugriff die Grenzen des abgebildeten Bereichs verlässt oder wenn ein Zugriff erfolgt, obwohl *mmap* nicht erfolgreich war. Die beiden Ausnahmenfälle sollten durch unterschiedliche Klassen repräsentiert werden, die einer gemeinsamen, von *std*::*exception* abgeleiteten Hierarchie angehören. Der Konstruktor selbst sollte aber keine Ausnahme auslösen. Stattdessen ist eine Testmethode anzubieten, mit der überprüft werden kann, ob die Abbildung (d.h. der *mmap*-Aufruf) erfolgreich war oder nicht. Erst wenn ein Zugriff erfolgt, sollte es ggf. zur Auslösung von Ausnahmen kommen. Die Signaturen aller Methoden der Klasse sollten explizit spezifizieren, ob sie potentiell Ausnahmen auslösen oder nicht.

Reizvoll wäre es, wenn bei einem Zugriffsversuch außerhalb des zulässigen Bereichs eine Fehlermeldung zu gewinnen wäre, die das präzise beschreibt. So könnte beispielsweise eine Meldung aussehen:

```
access violation: [510, 520) does not fit into [0, 512)
```

Wenn Sie das implementieren möchten, sollten Sie die Fehlermeldung bereits vollständig beim Konstruieren der Ausnahme erzeugen und in einem *std*::*string*-Objekt ablegen, das zu der Ausnahme gehört. In der *what*-Methode können Sie dann mit Hilfe der *c_str*-Methode die Zeichenkette als **const char*** zurückgeben. Wichtig ist hier, dass der Inhalt hinter dem Zeiger solange lebt wie das Ausnahmenobjekt selbst auch. Mit Hilfe eines *std*::*ostringstream*-Objekts können sie eine Zeichenkette befüllen:

```
std::ostringstream os;
os << ....;
message = os.str(); // message ist vom Typ std::string
```

Wie Sie den Zugriff auf den abgebildeten Speicher gewähren, ist Ihnen freigestellt. Es bietet sich aber an, den Zugriff für beliebige Typen zu erlauben. Wenn das als Template-Methode realisiert wird, wäre beispielsweise folgendes möglich für eine Datei, die zu Beginn einen binären Header hat, der durch den Datentyp *Header* repräsentiert wird:

```
MemObject mem{...};
if (mem.valid()) {
```

```
    Header& header{mem.access<Header>(/* position = */ 0)};
}
```

Zu beachten ist hier, dass der Template-Parameter bei dem Methodenaufruf von *access* notwendig ist, da der Rückgabetyp alleine nicht genügt, um den Typparameter automatisch herzuleiten.

Ergänzen Sie Ihre *MemObject*-Klasse um ein einfaches Testprogramm, mit dem Sie insbesondere auch in der Lage sind, das Auslösen der Ausnahmen zu testen.

Sie können wie üblich Ihre Lösung wieder einreichen:

```
thales$ tar cvf MemObject.tar *.?pp Makefile
thales$ submit cpp 10 MemObject.tar
```

Viel Erfolg!